# ACE View — an ontology and rule editor based on controlled English

Kaarel Kaljurand
Institute of Computational Linguistics, University of Zurich
kalju@ifi.uzh.ch

## ABSTRACT

We describe the architecture of a novel ontology and rule editor ACE View. The goal of ACE View is to simplify viewing and editing expressive and syntactically complex OWL/SWRL knowledge bases by making most of the interaction with the knowledge base happen via Attempto Controlled English (ACE). This makes ACE View radically different from current OWL/SWRL editors which are based on formal logic syntaxes and general purpose graphical user interface widgets.[1]

## Keywords

knowledge engineering, natural language based user interfaces, Attempto Controlled English, OWL, Protégé, ACE View

## 1. INTRODUCTION

We describe the architecture of a novel ontology and rule editor ACE View[2]. The goal of ACE View is to simplify the exploration and editing of expressive and syntactically complex OWL 2 [5] ontologies and SWRL [3] rulesets by basing the user interface on Attempto Controlled English (ACE) [1]. This makes ACE View radically different from current OWL/SWRL editors which are based on formal logic syntaxes and general purpose graphical user interface widgets (checkboxes, trees, etc.), and which are often seen as too complicated and confusing for domain experts with no background in formal methods [2]. ACE View integrates two mappings, ACE→OWL/SWRL and OWL→ACE, and is implemented as a plug-in for Protégé 4[3]. ACE View is currently available in binary form. We are working towards releasing it under an open source license.

The emerging OWL 2 specification describes several serialization syntaxes for OWL ontologies (RDF and XML based, functional-style, Manchester OWL Syntax). Some of these syntaxes are oriented towards machines and are thus inherently difficult to read and write for humans. Others have been designed for logicians and programmers, but lack the features that would bring OWL closer to domain experts who are often not well-trained in formal methods. E.g. [6] lists the problems that users encounter when working with OWL and expresses the need for a "pedantic but explicit" paraphrase language. While some of the problems are purely semantic (e.g. caused by misunderstanding the open world reasoning and the unique name assumption) and would be encountered in any syntax, many problems are rooted in the nature of current OWL syntaxes. Furthermore, many knowledge bases require a rule component, often expressed in SWRL. The proposed SWRL syntax, however, is completely different from the OWL syntaxes (mainly because it explicitly uses variables) even though there is an overlap of the semantics of OWL and SWRL. Query languages for OWL ontologies introduce yet another set of syntaxes.

The syntactic complexity can be hidden to some extent by front-end tools such as Protégé which use forms and other standard user interface widgets to support viewing and editing knowledge bases. Still, the relative richness of OWL and related languages means that for more complex expressions (negation, property restrictions, etc.), the user interface has to fall back to one of the standard syntaxes.

## 2. ACE ⇔ OWL/SWRL

ACE is a subset of English, such that each sentence in the chosen subset is interpreted unambiguously, relating the sentence to a unique logical form. The intention behind the design of ACE is to offer expressivity required in knowledge engineering tasks, but also remain a natural subset of English. The design minimizes what the users need to learn (assuming knowledge of English) to correctly compose ACE sentences and understand their meaning, e.g. to predict the paraphrases or the logical entailments of the sentences. Recently, ACE has been used in several Semantic Web projects, e.g. as an interface language in the semantic wiki AceWiki[4].

In order to make ACE interoperable with some of the existing Semantic Web languages, mappings have been developed to relate ACE to OWL, SWRL, and DL-Query (see a detailed description in [4]). For example, the mapping of ACE to OWL/SWRL translates the ACE text

> Everybody that does not own a car owns a bike.
> Every man that owns a car likes the car.
> Which car does John own?

into a combination of OWL axiom, SWRL rule and DL-Query (an OWL class expression).

$$\top \sqcap \neg (\exists\ own\ car) \sqsubseteq \exists\ own\ bike$$
$$man(?x) \wedge own(?x, ?y) \wedge car(?y) \rightarrow like(?x, ?y)$$
$$car \sqcap \exists\ own^{-}\ \{John\}$$

[1]This research has been funded by the European Commission and by the Swiss State Secretariat for Education and Research within the 6th Framework Programme project REWERSE number 506779. The author is currently supported by the Swiss National Science Foundation (grant 100014-118396/1). The author would like to thank Norbert E. Fuchs, Tobias Kuhn and Fabio Rinaldi for their valuable feedback.

[2]http://www.cl.uzh.ch/kalju/ACE_View/

[3]http://www.co-ode.org/downloads/protege-x/

[4]http://attempto.ifi.uzh.ch/acewiki/

The OWL→ACE mapping, on the other hand, allows us to verbalize existing OWL ontologies as ACE texts. This mapping is not just the reverse of the ACE→OWL as it also covers OWL axiom and expression types that the ACE→OWL mapping does not generate. For example, the OWL axiom

```
PropertyDomain(ObjectProperty(write) Class(author))
```

is verbalized as "Everything that writes something is an author.".

The mappings between ACE and OWL/SWRL provide an alternative syntax for OWL and SWRL. This syntax is readable as standard English and provides linguistically motivated syntactic sugar. It also makes the difference between OWL, SWRL and DL-Query invisible. This syntax is mainly intended for structurally and semantically complex OWL/SWRL knowledge bases for which visual methods and traditional syntaxes fail to provide a user-friendly front-end.

# 3. ACE VIEW

The ACE View editor lets the user manage an ACE text. An ACE text is a set of ACE snippets where each snippet is a sequence of one or more anaphorically linked ACE sentences. When a snippet is added to the text, it is automatically parsed and translated into OWL/SWRL. If the translation fails then the snippet is still accepted, it simply does not have any logical axioms attached and thus cannot participate in reasoning. In case the translation succeeds, the snippet is mapped to one or more OWL axioms and SWRL rules which are merged into the Protégé managed ontology. In case a snippet is deleted, its corresponding axioms (if present) are removed from the underlying ontology.

Alternatively, the ACE View user can switch to one of the standard Protégé views (or views offered by other Protégé plug-ins) to perform an ontology editing task. In case an OWL axiom is added in another view, then it is automatically verbalized and merged into the ACE text. If the verbalization fails (e.g. the verbalizer does not support the *FunctionalProperty*-axiom with data properties) then an error message is stored and the axiom is preserved in the ACE text in Manchester OWL Syntax. In case an axiom is deleted, then its corresponding snippet is deleted as well.

The ACE text (and thus the ontology) can be viewed and edited at several levels. **Word level** provides an access to individual words in the text so that the surface forms (singular, plural, past participle) of the words can be edited. As words correspond directly to OWL entities, they can be further annotated using the standard Protégé views. **Snippet level** provides access to three categories of snippets: asserted declarative snippets, asserted interrogative snippets (i.e. questions) and entailed (declarative) snippets. Asserted snippets are editable and provide access to their details (e.g. syntax errors, syntax-aware layout, ACE paraphrase, corresponding OWL/SWRL axioms). Questions are managed in the same way as asserted snippets but they additionally provide answers (sets of words) whenever the user runs the reasoner over the ontology. Entailed snippets are declarative snippets that follow logically from the ACE text. Similarly to questions, the set of entailments is updated whenever the reasoner is executed. Each entailment can be explored to find out the reason for the entailment (which is presented as a list of asserted snippets that cause the entailment). A basic mechanism for tracking changes in the entailment set is also provided. **Vocabulary** is a set of ACE content words. It can be sorted alphabetically or by frequency of usage, and standard Protégé views offer even more presentation options, e.g. the "back-bone hierarchy" of subclass and "part of" relations. The vocabulary level provides an easy access to the word level, each selected/searched word can be automatically shown in the word level, or its corresponding snippets in the text level. An **ACE text** is a set of ACE snippets. This set can be filtered, sorted, and searched. Reasoning can be performed on the whole text to find out about its (in)consistency. Linguistic metrics characterize the complete text (by number of sentences, number of sentences that map to SWRL, etc.).

ACE View is tightly integrated into the Protégé framework. Most operations done via standard Protégé menus (e.g. undo/redo, renaming) trigger a corresponding change in the ACE text. Any Protégé-supported reasoner can be used for semantic feedback, and any OWL/SWRL serialization syntax can be used to store the ACE text — the ACE snippets are saved as axiom annotations, and the wordform mappings as entity annotations. The main user interface element in ACE View is a table that organizes ACE snippets or words. The tables respond to Protégé events (e.g. selection of an entity) via filtering and highlighting.

# 4. CONCLUSIONS

ACE View introduces a novel paradigm to OWL/SWRL engineering. We assume that ontologies and rulesets are usually first expressed (in the minds of domain experts) in natural language, and thus working in ACE involves fewer conceptual problems. On the other hand, the combination of natural language based ontology editing and the standard form/formula-based editing offers more alternatives for the user and can result in an interesting synergy especially in the case of novice ontology engineers and domain experts working with semantically expressive and syntactically complex knowledge bases.

# 5. REFERENCES


[1] Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Attempto Controlled English for Knowledge Representation. In *Reasoning Web, Fourth International Summer School 2008*, 2008.

[2] Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a Controlled Natural Language for Authoring Ontologies. In *ESWC 2008*, 2008.

[3] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. Technical report, 2004.

[4] Kaarel Kaljurand. *Attempto Controlled English as a Semantic Web Language*. PhD thesis, Faculty of Mathematics and Computer Science, University of Tartu, 2007.

[5] Boris Motik, Peter F. Patel-Schneider, and Ian Horrocks. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft 11 April 2008. Technical report, 2008.

[6] Alan L. Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In *EKAW 2004*, 2004.